A Computer Aided Design Methodology For Printed Circuit Boards

Linda G. Bushnell and Vason P. Srini

Computer Science Division, EECS

University of California, Berkeley, CA 94720

ABSTRACT

We present a design methodology for realizing VLSI printed circuit board (PCB) designs using Computer Aided Design (CAD) tools on powerful workstations. The workstation and CAD tools allow one to create a gate-level design using TTL parts and interconnecting wires, simulate the functionality and timing of the gate-level design, package the design's TTL parts into PCB parts, place and route the PCB, create manufacturing data, and simulate the design at the board-level. A design example, the VLSI-PLM PC Board being developed at the University of California at Berkeley, is given to illustrate this procedure. The VLSI-PLM PC Board is a processor board for the VLSI-PLM Chip [SRINI], which is a high performance CMOS processor for executing computer programs written in the Prolog language.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JAN 1989		**00-00-1989 to 00-00-1989**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Computer Aided Design Methodology For Printed Circuit Boards	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Department of Electrical Engineering and Computer Sciences,Berkeley,CA,94720**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We present a design methodology for realizing VLSI printed circuit board (PCB) designs using Computer Aided Design (CAD) tools on powerful workstations. The workstation and CAD tools allow one to create a gate-level design using TTL parts and interconnecting wires, simulate the functionality and timing of the gate-level design, package the design's TTL parts into PCB parts, place and route the PCB, create manufacturing data, and simulate the design at the board-level. A design example, the VLSI-PLM PC Board being developed at the University of California at Berkeley, is given to illustrate this procedure. The VLSI-PLM PC Board is a processor board for the VLSI-PLM Chip [SRINI], which is a high performance CMOS processor for executing computer programs written in the Prolog language.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**38**	

Table of Contents

List of Figures

1. Introduction

In the area of computer architecture, one not only wants to develop high performance architectures, but also to have an efficient and accurate development cycle. In order to guarantee a flawless PCB design before fabrication, the current trend is to use available software packages on computer workstations. By using these CAD tools, the designer can create the computer architecture gate-level design using TTL gates and interconnecting wires. One can quickly verify the design by performing gate-level functional and timing simulations. It is at this step where most of the design flaws are found and corrected. The gate-level design can then be developed into a board-level design through a series of steps involving packaging the TTL gates into PCB chips, placing these chips on the PCB, and routing the board. The designed PCB can then be verified with simulations. The final product of this design process is the manufacturing data, e.g., drilling and milling information, Gerber artwork, and assembly and fabrication drawings. This manufacturing data can be directly used to fabricate a PCB. Through each step, the designer can check and correct any error easily with the CAD tools.

This paper covers two main topics. First we discuss the procedure of realizing PCB designs using Mentor Graphics CAD tools on Apollo workstations. The seven major steps in this procedure are presented in detail. The second topic is an illustration of this design process. We present a complete example: the VLSI-PLM PC Board currently under development at the University of California at Berkeley. The VLSI-PLM PC Board is a processor board for the VLSI-PLM Chip [SRINI], which is a high performance CMOS processor for the Prolog computer language.

Section 2 discusses the statement of the problem. Section 3 provides a detailed description of the method of realizing a board using CAD tools. Section 4 presents an actual PCB designed using this methodology. Section 5 discusses the problems encountered in the design procedures and the limitations of the CAD tools. Section 6 summarizes with a conclusion.

2. Problem Scope

When one has an idea for a PCB design, the corresponding gate-level design can be drafted with pencil and paper. This drafted design can then be fabricated into a wire-wrapped board. However, it is more efficient to create both the gate-level and board-level designs on a computer workstation by using CAD tools. All of the design modifications, simulations, and verifications can be performed more

efficiently on computers than on wire-wrapped boards. The final product of using the CAD tools, the manufacturing data, can then be used to fabricate a PCB.

The first step in the procedure to design a PCB using CAD tools is to put the gate-level schematic on the computer by using standard TTL library parts, custom designed parts, and interconnecting wires. This design schematic can then be checked for errors and corrected. Once the design schematics are flawless, a simulation can be performed to verify the design functionality and timing. If any problem is found, it is easy to modify the original design schematics and re-run the simulations. By using this iterative procedure, one can produce an accurate gate-level design in a relatively short amount of time.

The next task is to package the TTL gates in the gate-level schematic into PCB chips that can be used on a software modeled PCB. In this step, the design process can be shortened by using standard PCB geometries provided in the libraries of the CAD software. Custom-made geometries can also be easily constructed.

These packaged parts are then placed on the PCB by using the interactive and automatic PCB chip placing CAD tools. The placement can also be checked and improved. The next step is to route the board by using the automatic CAD routers. Some interactive routing may have to be done for complicated designs.

The manufacturing data can be created and sent out for fabrication. The CAD tool for producing manufacturing data allows the designer to produce drilling, milling, Gerber artwork, and assembly and fabrication drawings.

The final step in the design process is to perform board-level simulations. These simulation results can be compared with the gate-level simulation results and necessary design changes can be made so that they will agree.

3. The Design Methodology

In this section, we describe in detail a step-by-step procedure to realize complex PCB designs. There are seven major steps in this procedure. The CAD tools mentioned are from Mentor Graphics (MG) version 6.1 software [MENTOR], and are installed on Apollo workstations, under version 9.7.0.4 Aegis operating system.

3.1 Schematic Capture

For the first step in the design process, the formulated design is drawn on the computer workstation using the MG Idea Station schematic capture tools *Neted* and *Symed. Neted* allows a multi-level hierarchical design to be drawn on a schematic sheet using TTL standard parts, custom-made parts, and interconnecting wires. The interconnecting wires are also called "nets", for which this tool is named. A hierarchical design implies a three-dimensional drawing. *Symed* is used to make the custom parts' symbols and any component symbols that will be placed on the Neted schematic sheets of the design. The symbols usually have their own schematic sheets (made with Neted) underneath them. In this step, a multi-level hierarchical design can be made by using a top-down approach. The first step is to draw and edit a symbol for the entire part to be designed by using Symed. For example, *Figure 1* [MENTOR] shows the MG symbol for a D Flip-Flop.

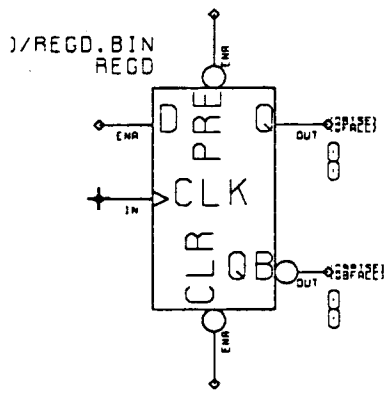

Figure 1 Sample Symed Drawing: The D Flip-Flop

Next, one makes a schematic sheet inside of this symbol, by using Neted, where the design schematic can be drawn. On this top-level sheet, TTL primitive component symbols and custom-made symbols can be connected using wires. The primitive component symbols are provided by the MG library. *Figure 2* [MENTOR] shows the MG Neted schematic for a D Flip-Flop.

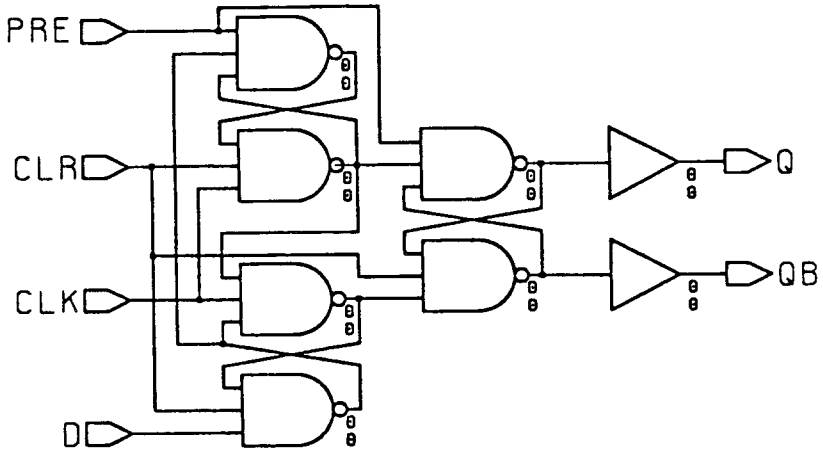

Figure 2 Sample Neted Drawing: The D Flip-Flop

The non-primitive symbols on this top-level sheet may have schematic sheets inside of them, thereby creating a multi-level hierarchical design. *Figure 3* [MENTOR] presents the concept of a hierarchical design. The design schematics must be checked for error upon completion. This checking of the design is performed automatically by the Neted tool. All errors have to be resolved before continuing in the design process.

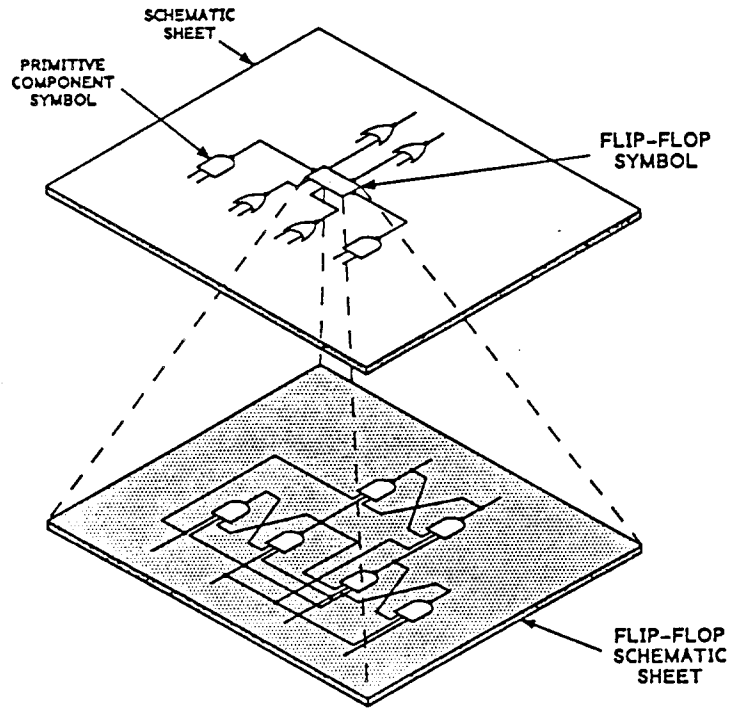

Figure 3 Hierarchical Designing

3.2 Gate-level Simulation

After completing the design schematics and checking them over, the next step is to transform the multi-level hierarchical design into a flat array of primitive elements consisting only of basic cells and wires. A basic cell is one which has associated with it a computer program describing its behavior, called a Behavior Language Model (BLM). A BLM is written in C or Pascal and models the functional behavior of a digital component. The transformation is easily accomplished by the MG tool *Expand*. Using this expanded design, the MG logic simulator tool *QuickSim* interactively allows for the verification of the functionality and timing of the design. The designer can establish a design cycle of applying stimulus to the design, running the simulation, analyzing the results, and then modifying the design based on these results. This iterative process, shown in *Figure 4*, will guarantee an accurate gate-level design.

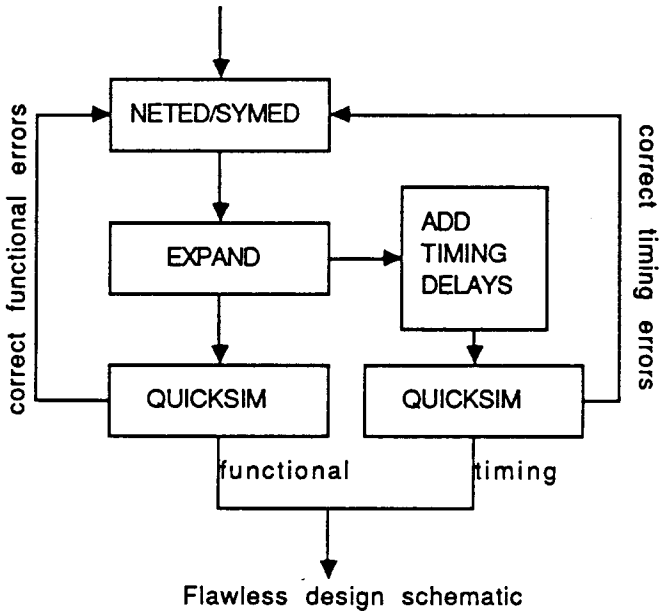

Figure 4 Gate-level Design Procedure

Before invoking QuickSim, macro files can be made to prepare an environment for testing and tell the simulator what tests to run. These macro files contain QuickSim commands that set signals to specified values at certain clock times. Many simulation runs can be incorporated into one macro file so that the part can be tested completely. This macro file can then be called from within QuickSim and the simulation will be done automatically. The output is an ascii file that contains the test pattern results.

For the functional simulations, QuickSim is used directly after Expand and the schematic capture. However, the procedure is a little more complicated for the timing simulations. To get a more accurate simulation of the effects of net delays in the design, the designer must add delays to the expanded design schematic. Based on the BLMs of the basic cells, a MG program automatically determines the capacitance and resistor values and sets up delay values for the various nodes and interconnecting wires. The simulation is then run using QuickSim, giving more realistic results. Please see the discussion in the section on board-level simulation for more detail.

3.3 Packaging the Parts

For this section through section 3.7 please reference *Figure 5* [MENTOR] to see the iterative procedure for creating PCBs. With a gate-level design schematic of correct functionality and timing, one can package the TTL gates into PCB chips to be placed on a PCB model. There are three steps to accomplish this. The first step is to eliminate the hierarchy in the schematic sheets to create a flattened version of the gate-level design, using the MG Board Station tool *Expand_PCB*. This tool is similar to Expand, but Expand_PCB also creates a logic database to be used by the MG PCB tools. This logic database contains the connectivity information extracted from the design schematic sheets.

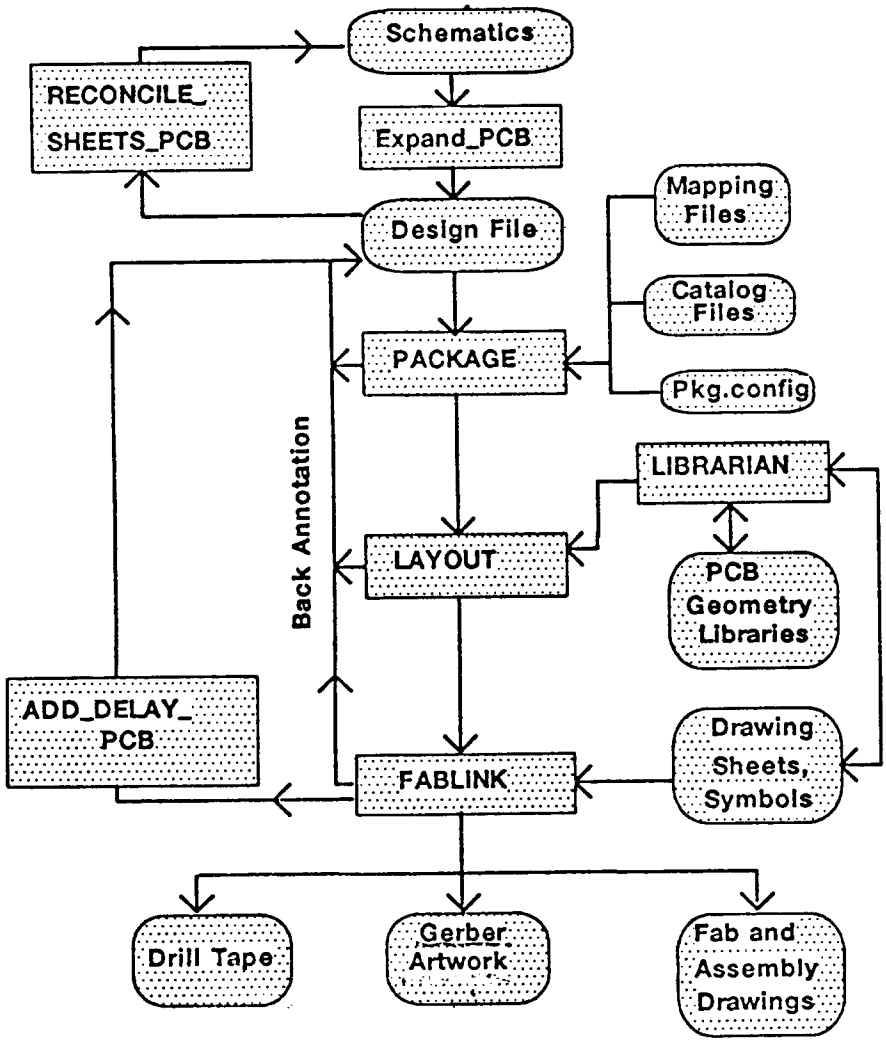

Figure 5 Board-level Design Procedure

The second step is to define and modify board-level parts for the design. The MG Board Station tool *Librarian* is used to build library parts such as component geometries, logos, drawings, and pad-stacks. It also is used to create catalog and mapping files for the component geometry parts. By using Librarian, one can also group together the parts necessary for the design into a data file (parts file) to be used in the PCB layout. These component geometries are used to represent the actual physical components of the design. The catalog and mapping files show the connection between symbol pins on the design schematic and PCB geometry component pins. In addition, the catalog and mapping files show which gates and pins can be swapped. Any PCB parts needed for the design that are not in the software libraries must be created with Librarian.

Eight types of parts can be made using Librarian: (1) artwork stack-ups, which associate the data created on a board layer to a sheet of film; (2) PC Boards; (3) component geometries; (4) mechanical or assembly drawings; (5) manufacturing panels, which show the arrangement for fabricating more than one board part per manufacturing operation; (6) pin pad-stacks, which describe the connection of a component pin on each PCB layer; (7) via pad-stacks, which are plated through-holes on each PCB layer; and (8) generic parts.

The third step is to assign symbols from the design schematic to PCB components by using the MG Board Station tool *Package*. These PCB components are to be placed on the PCB model. The packaging procedure can be done automatically or interactively. Package also checks the catalog and mapping files for errors, which must be corrected before proceeding to the board layout.

3.4 Placing the Parts

At this stage in the design methodology, the design has been transformed into many PCB chips. These components can be placed on the top and/or bottom surfaces of the PCB by using the MG Board Station tool *Layout*. One can place certain parts first then let the automatic layout tool place the rest. The placement of the chips on the PCB can be improved by a built-in algorithm in Layout. The placement must also be checked when completed. The designer can choose the number of signal and power layers for the layout and routing of the board.

3.5 Routing the Board

When all of the components are placed on the PCB, the next step is to route the wire connections between these components, which is performed by using Layout. This procedure can also be done automatically by a built-in routing algorithm or interactively. The best procedure is to route with the following built-in algorithms: (1) Pattern for eight passes, (2) Auto for five passes, (3) Manufacturing for two passes, and (4) Auto for five more passes. If the board is not 100% routed after these twenty passes, the remaining wires must be manually routed by the designer. One can try to use the automatic router again to complete the few remaining wires, but it is much quicker to route them yourself. After having a completely routed board, the wires can be checked by a built-in checking algorithm. Any error must be fixed by the designer.

3.6 Producing Manufacturing Data

The last step in the CAD procedure for designing a PCB is to generate the manufacturing data. The MG Board Station tool *Fablink* allows one to produce photoplotter data (Gerber artwork), drilling and milling data, and fabrication and assembly drawings. The output from this step can be used to manufacture a PCB. First the designer must define the aperture settings. The shape (circle or rectangle), diameter, power, and type (flash, trace, or both) must be specified. Next the Gerber artwork is created automatically. Each artwork shows one layer of the PCB. One artwork drawing is of the silk screen layer of the PCB, while the additional artwork drawings correspond to the signal and power layers. The drill magazine information must be entered by the designer. The drill size, upper bound, lower bound, and whether or not it is to be plated must be specified. The drill output can then be created automatically. This drill representation can be viewed and edited. Most of the milling data must be created by the designer. The assembly and fabrication drawings are made automatically and the designer can edit them to add text and dimensions.

3.7 Board-level Simulation

In order to ensure that the functionality and timing of the PCB is correct, the designer must perform simulations at the board-level. The design file must be back-annotated to allow for a more accurate simulation of the effects of the net delays in the board. *Add Delay PCB* is the MG software tool that performs the delay annotation. This program modifies the rise and fall times in the design specifications to reflect any changes in delay due to net routing and operating voltage and temperature. It also reflects the designer's choice of best, nominal, and worst case analysis.

Add Delay PCB will calculate the delay for the following cases: (1) fanout only delay, in which only loading capacitances are added to the net delays, (2) layout only delay, in which only the delay due to net length is added to the net delays, (3) expected layout delay, in which the expected delay due to layout is calculated (based on the net loading) and added to the net delays, (4) fanout and layout delay, in which both the layout and fanout delays are calculated and added to the net delays, and (5) fanout and expected layout delay, in which the expected layout and fanout delays are calculated and added to the net delays. The delay algorithm is designed for two layer metal CMOS gate arrays and considers capacitance delays due to fanout and interconnective distances. The algorithm calculates delays on a per net basis in

which specific pin-to-pin delays are not considered. The values for the resistances and capacitances are estimated by the algorithm. However, real values will be used after the actual wire-wrapped board is completed. After inserting the appropriate delays, the tool QuickSim can be used to simulate the design. The board-level simulation results should be compared to the gate-level simulation results. The designer can then make necessary design modifications if any discrepancies are found.

4. Example: VLSI-PLM PC Board

We now present a complete example: the VLSI-PLM PC Board under development at the University of California at Berkeley. There are two purposes of this example for this paper. The first is to show an actual application of the CAD methodology described above. The second purpose it to describe the design of the VLSI-PLM PC board, a processor board for the VLSI-PLM Chip. The Chip is a CMOS processor for the Prolog computer language. The board consists of the VLSI-PLM Chip, Display Registers, Decoder, DB25 Connector, three 3x32 Edge Connectors, two 3x32 DIN Connectors, and a part for the interface logic. The PCB measures 36.6 cm by 21.9 cm and was designed after a VME packaging panel. There are a total of 106 PCB chips that cover the entire board. The Symed and Neted drawings for the whole board are in the appendix in *Figures 6 and 7*, respectively. The board is designed to be used in a Sun system or as a stand-alone, operating at a range of 4 to 20 MHz.

4.1 Display Registers

Display Registers will be used to test the PCB after fabrication. This part contains the logic that allows certain signals to be viewed by the test engineer on the Digital Analysis System (DAS). These signals are MEMDATBUS.P(31:0) and MARBUS.P(27:0) which are connected to the VLSI-PLM Chip and come from the interface logic. The schematic drawings are shown in *Figures 8 and 9* (appendix). This part was packaged into many MG library PCB geometries (DIPs) for the layout.

4.2 Decoder

Decoder will also be used in the testing of the fabricated VLSI-PLM PC Board. This part contains the logic that translates the tester panel signals into signals for the VLSI-PLM Chip and vice versa. Specifically, the tester signals are the read, write, address select, enable, and 8-bit data line. *Figures 10*

and 11 (appendix) show the schematic drawings for Decoder. This block was also packaged into many MG library PCB parts.

4.3 DB25

The DB25 is a 25-pin connector that connects the interface board to the tester panel. This will allow for the testing of the fabricated VLSI-PLM PC Board. See *Figure 12* (appendix) for the symbol of the DB25 connector. The PCB geometry for the DB25 connector was made with Librarian by taking a DIP20 geometry from the MG library, adding five more pins, and redrawing the body.

4.4 Edge Connectors

The three 3x32 Edge Connectors (*Figure 13* in appendix) are used to attach the fabricated VLSI-PLM PC Board to a backplane, if desired. The geometry for the Edge Connectors was provided by the MG library. Each PCB geometry contains three rows of 32 pins for a total of 96 pins.

4.5 DIN Connectors

The two DIN Connectors (*Figure 14* in appendix) are similarly used for attaching the fabricated VLSI-PLM PC Board to the backplane. The DIN Connectors contain 96 pins. The PCB geometry for the DIN Connectors was copied from the Edge Connector geometry.

4.6 VLSI-PLM Chip

A detailed description of this chip can be found in [SRINI]. The symbol is shown in *Figure 15* in the appendix. This chip is a high performance VLSI CMOS 32-bit microprocessor that executes Prolog programs with high regularity, low power dissipation, and a small instruction set. The chip has 120 pins and is 11 mm by 9.5 mm. The design of the VLSI-PLM PC Board only uses the symbol for its design. The underlying logic for the chip has yet to be made into a BLM, where a simplified version of the logic is encoded in the computer language Pascal. In this sense, the chip acts like a connector. The geometry for the VLSI-PLM Chip was made to the specifications of NTK division of NGK Spark Plug Company. This geometry is a 168 terminal LSI pin grid array, but only 120 pins are used by the chip, including 18 power and ground.

4.7 The Interface Logic

The core of the design of the VLSI-PLM PC Board is the interface logic, which prepares most of the important signals for the VLSI-PLM Chip. The interface logic schematics are shown in *Figures 16 and 17* in the appendix. This is a multi-level hierarchical design. This logic was packaged into many MG supplied PCB parts. We will briefly describe the design and construction of the interface logic, also known as the VLSI-PLM Board (VPB). See reference [Nguyen] for a detailed description. The main purpose of the VLSI-PLM Board is to demonstrate the performance of the VLSI-PLM Chip.

The VPB design contains twelve sub-blocks, which are as follows: (1) Clock Generator, (2) Pushbuttons, (3) Toggle, (4) Atomicb, (5) Mirsignals, (6) Marbuff, (7) Passmclk, (8) Opcodebus, (9) Forceaddr, (10) Xcver, (11) Memdatbus, and (12) the VLSI-PLM Chip. The VPB was designed using the above mentioned design methodology with the Mentor Graphics IDEA Station tools on Apollo workstations. The VPB was created as a multi-level hierarchical design, allowing one to better understand the functionality.

After all of the simulations were performed, the VPB was proven to be correct with respect to functional and timing specifications. The VPB was actually wire-wrapped by the Electronics Support Group of the Electrical Engineering and Computer Science Department (EECS) at the University of California at Berkeley. A micro-processor-controlled tester console was also developed to automate some of the testing, provide inputs to the board's switches, display desired results, and provide the board with desired constants. The wire-wrapped board was designed to be used in a Sun system or as a stand-alone panel, operating at a frequency of 4 to 20 MHz. The total wire-wrapped part occupies 18 cm by 22 cm in a board 40 cm by 36 cm, with a total of 95 IC chips including the VLSI-PLM Chip. This wire-wrapped board was tested using the Digital Analysis System. The testing consisted of checking for connectivity and functionality. The test results verified the simulations of the computer-designed interface logic.

4.8 Layout and Routing

The interface board layout is shown in *Figure 18*, and the routed board is shown in *Figure 19*, both in the appendix. The geometries for the various parts in this design can be seen in these figures. The three 3x32 Edge Connectors are along the top, the DB25 Connector is in the left bottom corner, and the

two DIN Connectors are to the right of the DB25. The VLSI-PLM Chip is placed above the DIN Connectors. The various DIP parts are the rest of the design's logic. The board outline geometry was designed using Librarian as a VME packaging panel part no. 8139-VME922-02X with dimensions 36.6 cm by 21.9 cm. Eight signal layers, one silk-screen layer, and eight power layers were used to layout and route the board. For the layout of the VLSI-PLM PCB, the three Edge Connectors, two DIN Connectors, DB25, and the VLSI-PLM Chip were placed first, then the rest of the components were placed automatically. When the layout was finished, we made one pass through the improve placement algorithm on the automatically placed parts.

After completing a good layout, the board was automatically routed with three different algorithms with a total of 20 passes. We used the Pattern algorithm eight times, the Auto algorithm five times, the Manufacturing algorithm two times, followed by the Auto algorithm again for five passes. The board has a total of 106 PCB parts, 4259 segments, 152 vias, no fill areas, and 405 nets which covering the entire board.

4.9 Manufacturing Data

The Gerber artwork was created by first defining an aperture settings for the artwork. The aperture settings will be used by the photoplotter in the fabrication of the VLSI-PLM PC Board. There were seventeen artwork layers, corresponding to the layout layers, produced automatically by the CAD software. One is able to view, check, and possibly edit each artwork layer. A sample of the Gerber artwork produced is shown in *Figures 20-22* in the appendix. The first figure is of the silk-screen layer. *Figures 21 and 22* are of the first and second signal layers, respectively.

The fabrication and assembly drawings are in the appendix in *Figures 23 and 24*, respectively. These drawings were made by first specifying the drill size for the design. For this design we used a plated drill with a hole size of 0.0280. The drill was produced automatically. The assembly drawing was then generated automatically and the dimensions and comments were added manually. The fabrication drawing was also made automatically, with the drill table, comments, and dimensions being added by the designer. For the fabrication drawing, one has the option to make the drill holes visible, which was done for this design. No milling data was created. This is the final product that will be used to fabricate the VLSI-PLM PC Board.

4.10 Board-level Simulation

No board-level simulations were performed on the VLSI-PLM PC Board since we have not yet written a BLM for the VLSI-PLM Chip. We have to come up with a simplified model of the Chip, for the actual logic would be too complicated for a BLM.

5. Problems and Limitations

In this section we will describe some of the problems encountered during the design process and the limitations of the CAD tools. One problem was learning how to use the Mentor Graphics Boardstation tools from the documentation. It took a long time to learn that the "parts" file used in Boardstation must be made with Librarian. We used the MG hot-line often for help in many software problems that could not be solved by referring to the documentation. Another time consuming task was to make the mapping files (used in Package) from all of the existing component libraries (F, LS, general). Most of the mapping files had to be edited for the correct information. The catalog files also had to be fixed to work correctly in Package.

One limitation to creating a design using the MG tools is that often a part needed for schematics does not exist in the MG component libraries. Either a BLM of the part must be created or other parts must be used. It was also time consuming to create the custom-made geometries needed for the PCB design using Librarian. It was difficult to make sure all geometries had the correct properties attached to them so that the Layout tool worked correctly. Our design time was often hampered by system administration work, such as the acquisition of new authorization codes from MG, Apollo system maintenance, MG software updates, and installation of new Apollo DN3000 and DN4000 computers.

6. Conclusion

In this paper, we have described a procedure to realize a VLSI PCB design using CAD tools on computer workstations. Creating the gate-level design on the computer workstation, simulating the functionality and timing of the gate-level design, packaging the design's TTL parts into PCB parts, placing and routing the PCB, creating manufacturing data, and PCB-level simulations were discussed. We illustrated the CAD design process with a complete example, the VLSI-PLM PC Board. The design of the VLSI-PLM PC Board was presented in detail. After overcoming many problems and limitations,

we found the CAD methodology to be very efficient in our design of the VLSI-PLM PC Board. The VLSI-PLM PC Board was designed to be 36.6 cm by 21.9 cm with the whole area taken up by the board's 106 PCB chips. The interface logic was actually constructed into a wire-wrapped board by the Electronics Research Lab of the EECS Department. For comparison, the wire-wrapped VLSI-PLM Board is 40 cm by 36 cm with 95 chips which cover 18 cm by 22 cm of the board.

Future work on the VLSI-PLM PC Board includes writing an efficient BLM for the VLSI-PLM Chip, performing board-level simulations with this BLM, analyzing the simulation results, and comparing the results to the DAS simulation results from the actual wire-wrapped board. After these procedures are completed, the PCB manufacturing data can be used to fabricate the VLSI-PLM PC Board.

Acknowledgement

The VLSI-PLM Chip design was developed in the Aquarius project in the CS Division at the University of California in Berkeley by a research team headed by Dr. Vason P. Srini. The wire-wrapped version of the interface logic of the VLSI-PLM PC Board was designed by Dr. Srini and Lau T. Nguyen, implemented and tested by Lau, Joao Wentcovitch, and Katherina Law. We are thankful to the members of the Aquarius project, especially Tam Nguyen, for their help, comments, and suggestions.

This work was partially funded by the Defense Advance Research Projects Agency (DOD) and monitored by Naval Electronics System Command under contract No. N00039-84-C-0089, NCR Corporation, Dayton, Ohio, and National Science Foundation. Equipment and other support for the project has been provided by DEC, NCR, Apollo, ESL, and Xenologic.

References

1. Mentor Graphics Corporation Software Documentation, Beaverton, Oregon, 1988.

2. Srini V.P., J.V. Tam, T.M. Nguyen, B.K. Holmer, Y.N. Patt, A.M. Despain, *Design and Implementation of a CMOS Chip for Prolog*, Technical Report No. UCB/CSD 88/412, Computer Science Division, EECS, University of California, Berkeley, California, March 1988.

3. Nguyen, L.T., L.G. Bushnell, and V.P. Srini, *The VLSI-PLM Board: Design, Construction, and Testing,* Technical Report No. UCB/CSD 89/*, Computer Science Division, EECS, University of California, Berkeley, California, January 1989.

* To be assigned

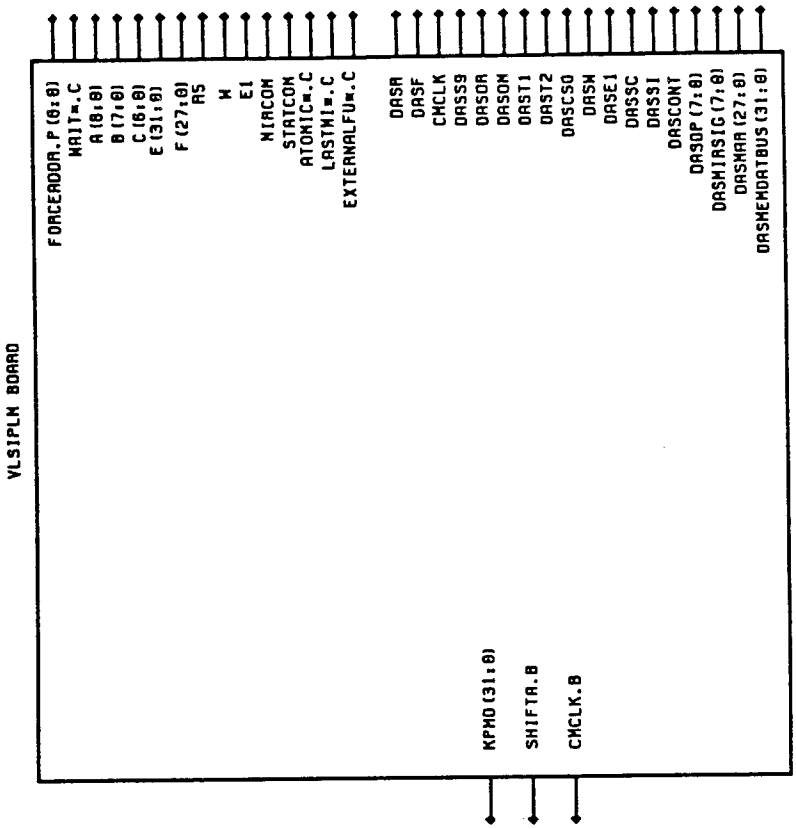

Figure 6 VLSI-PLM PC Board Symbol

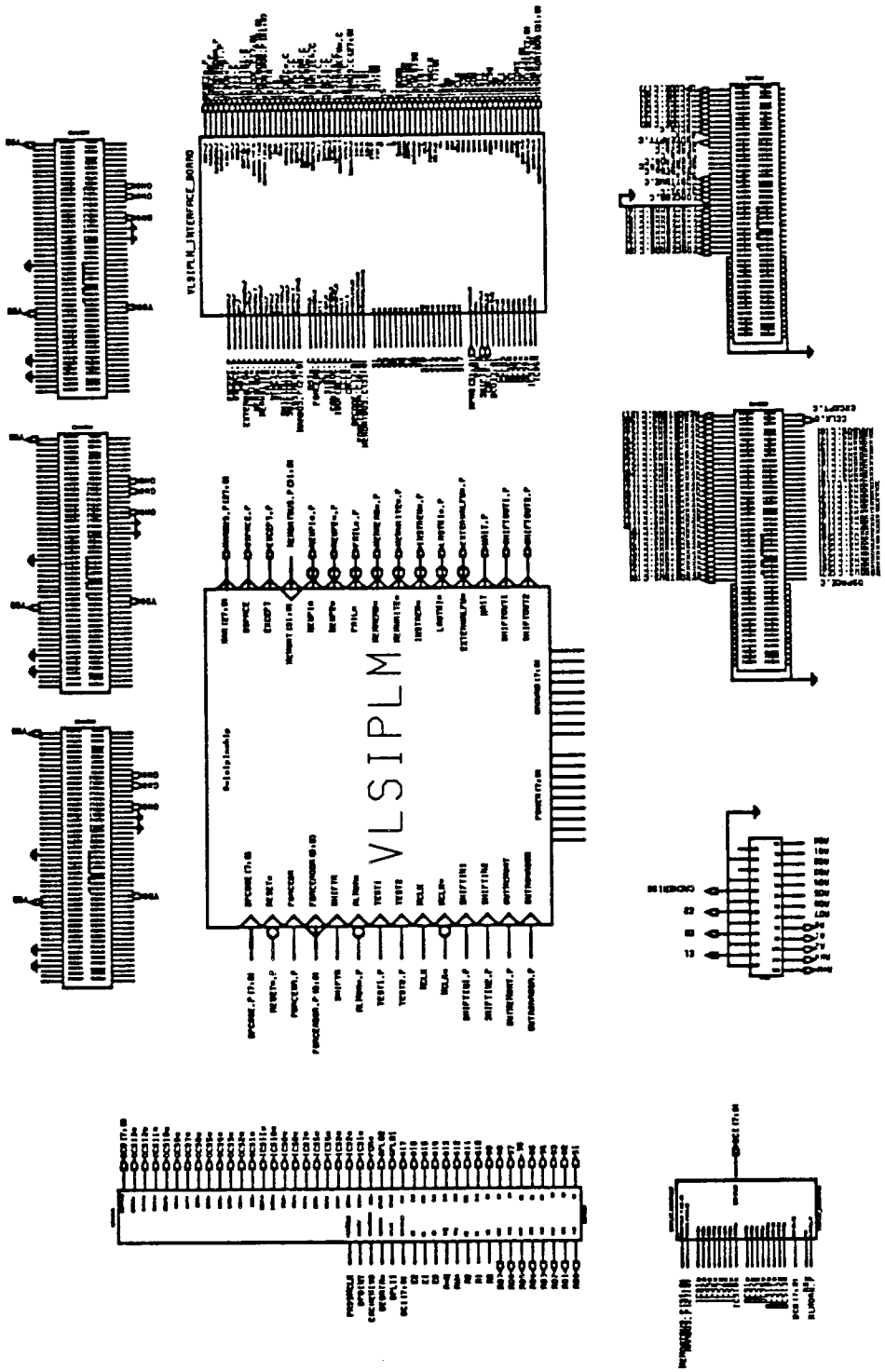

Figure 7 VLSI-PLM PC Board Schematic

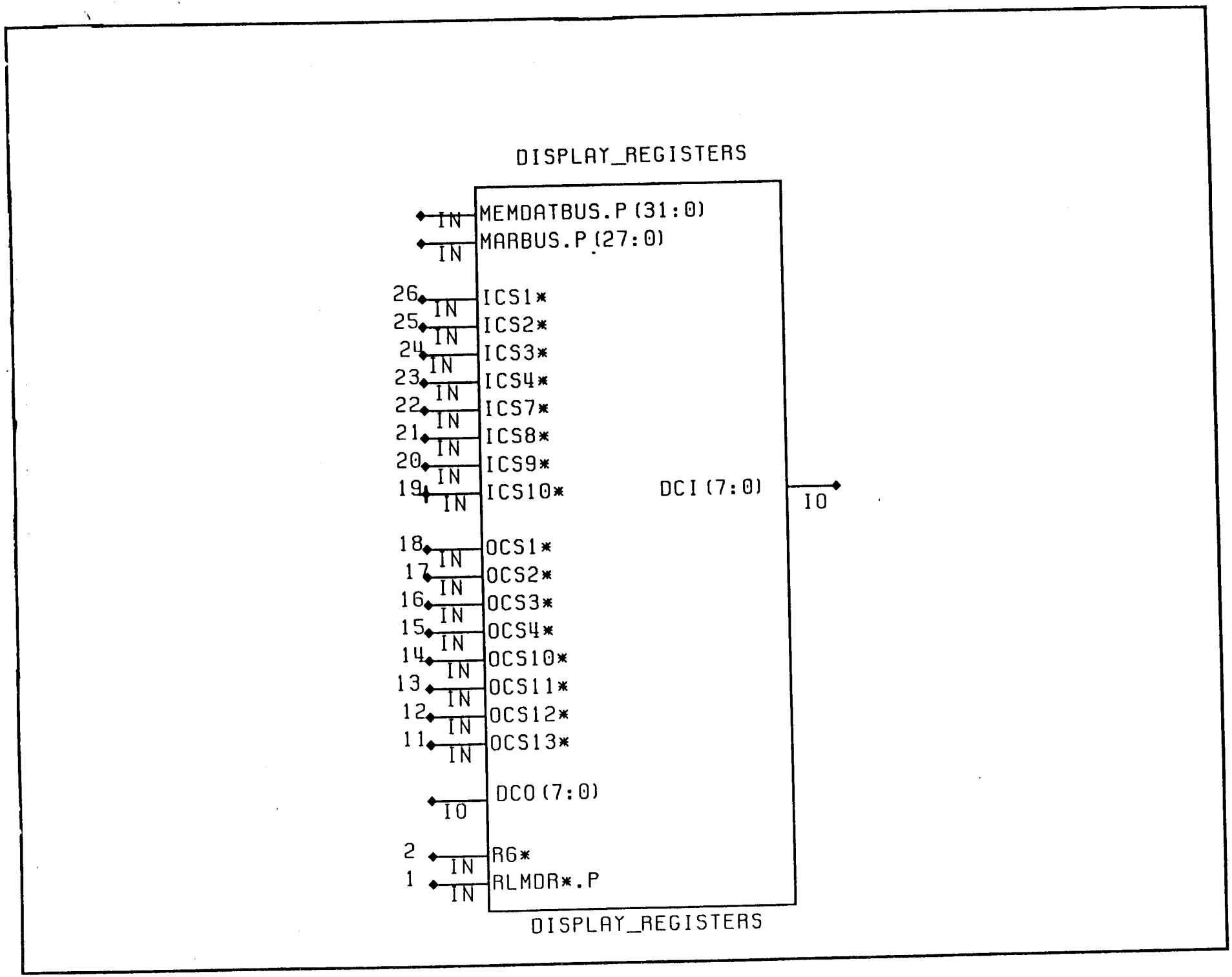

Figure 8 Display Register Symbol

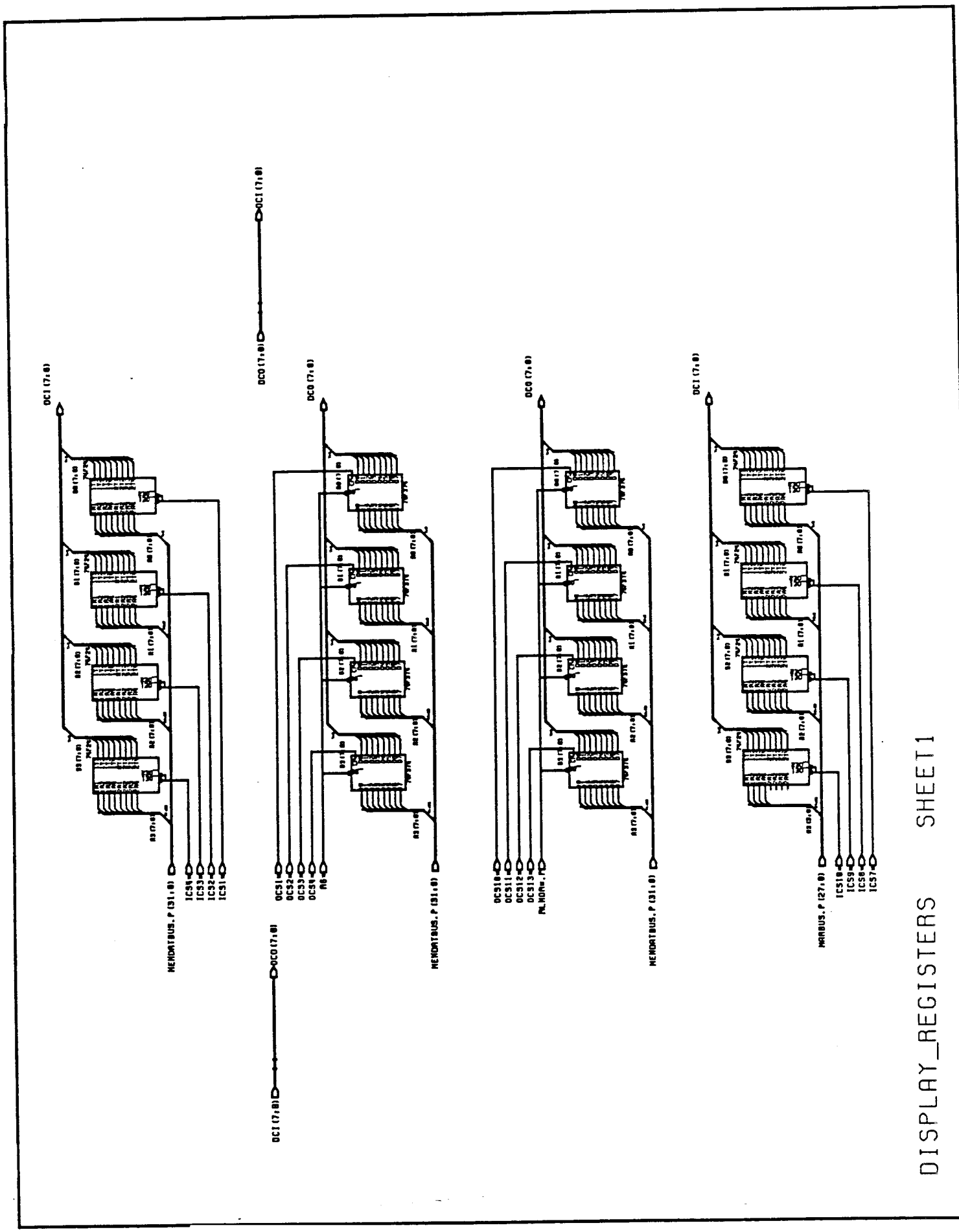

Figure 9 Display Registers Schematic

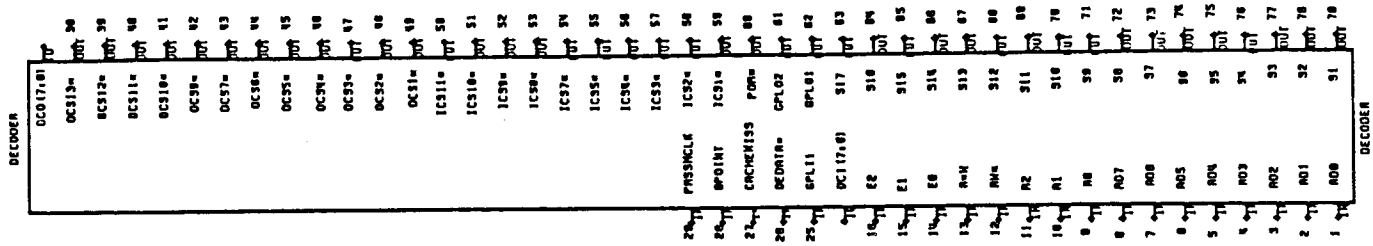

Figure 10 Decoder Symbol

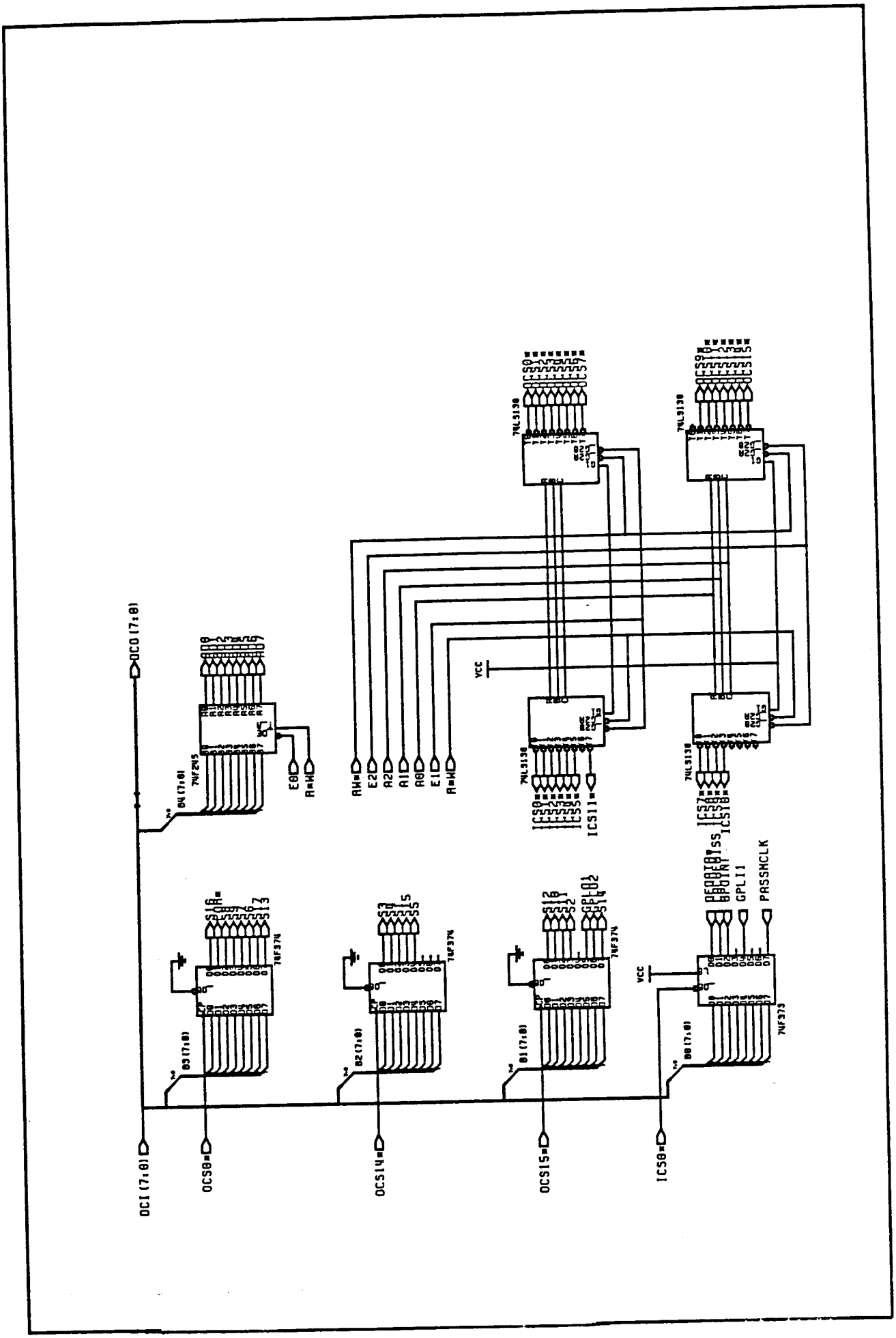

Figure 11 Decoder Schematic

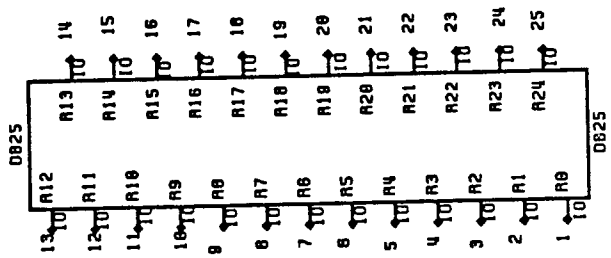

Figure 12 DB25 Symbol

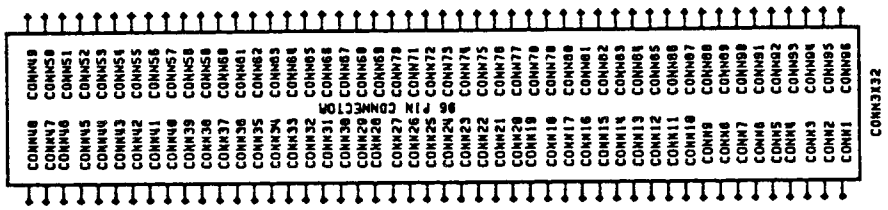

Figure 13 Edge Connector Symbol

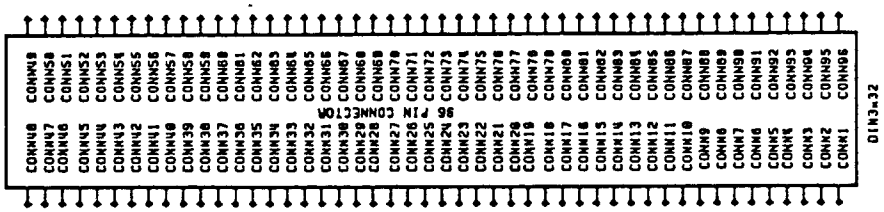

Figure 14 DIN Connector Symbol

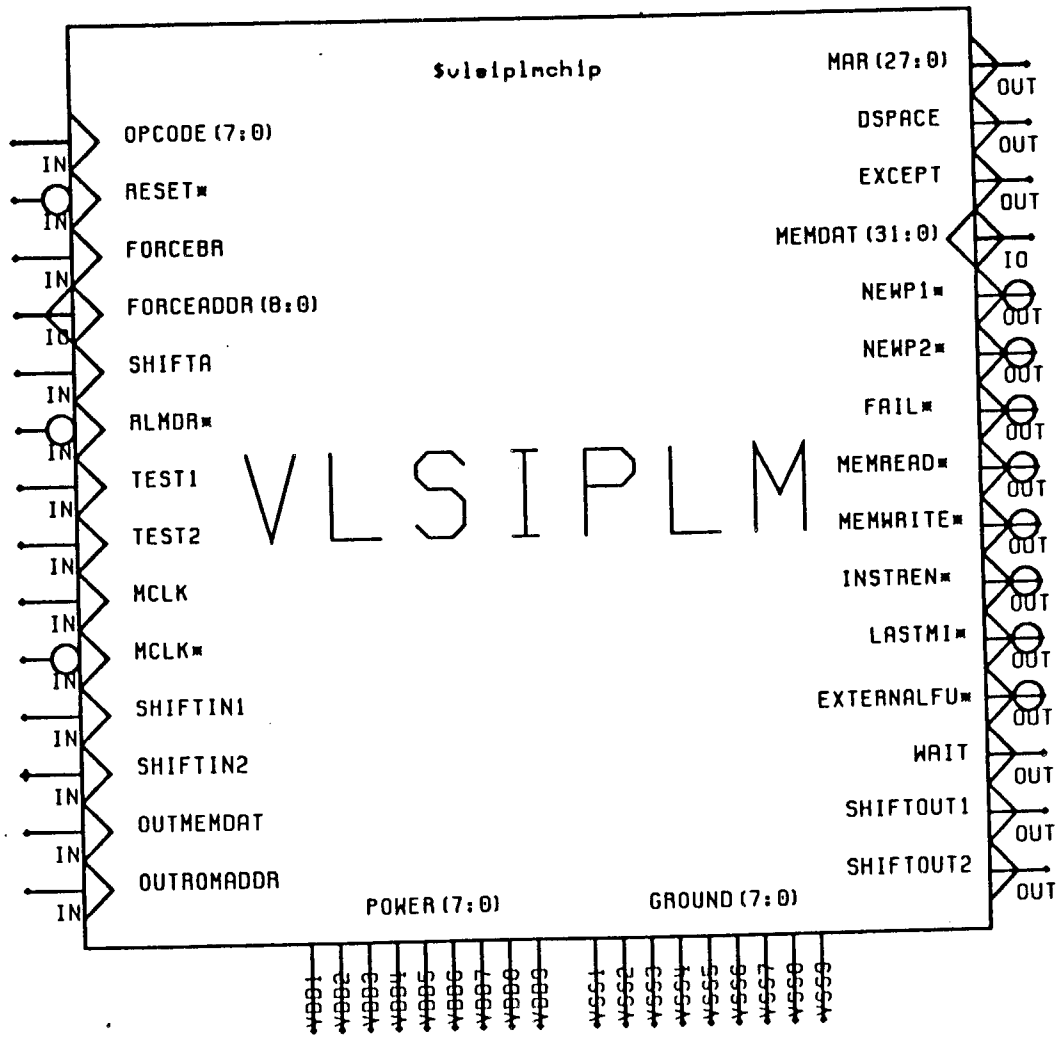

Figure 15 VLSI-PLM Chip Symbol

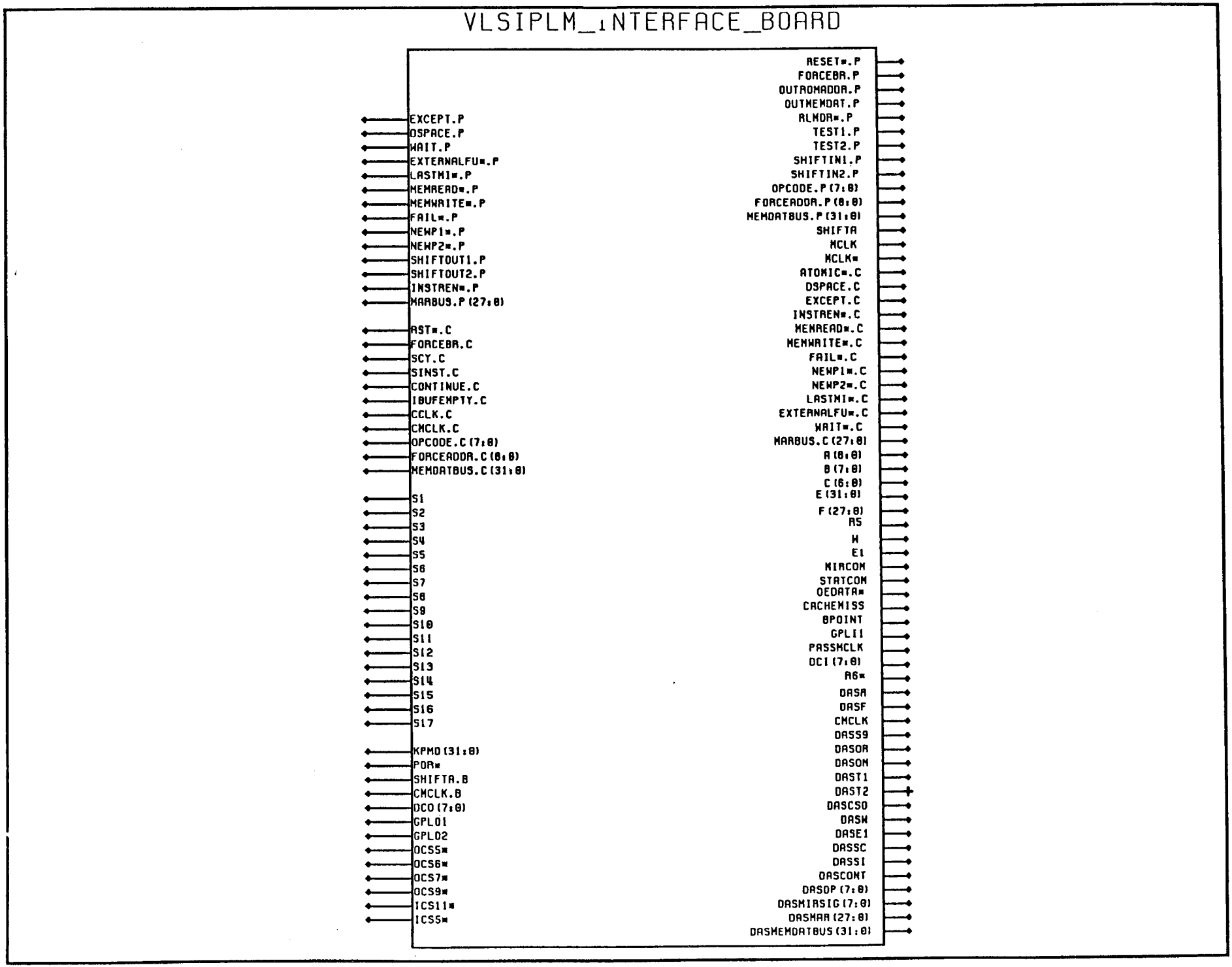

Figure 16 Interface Logic Symbol

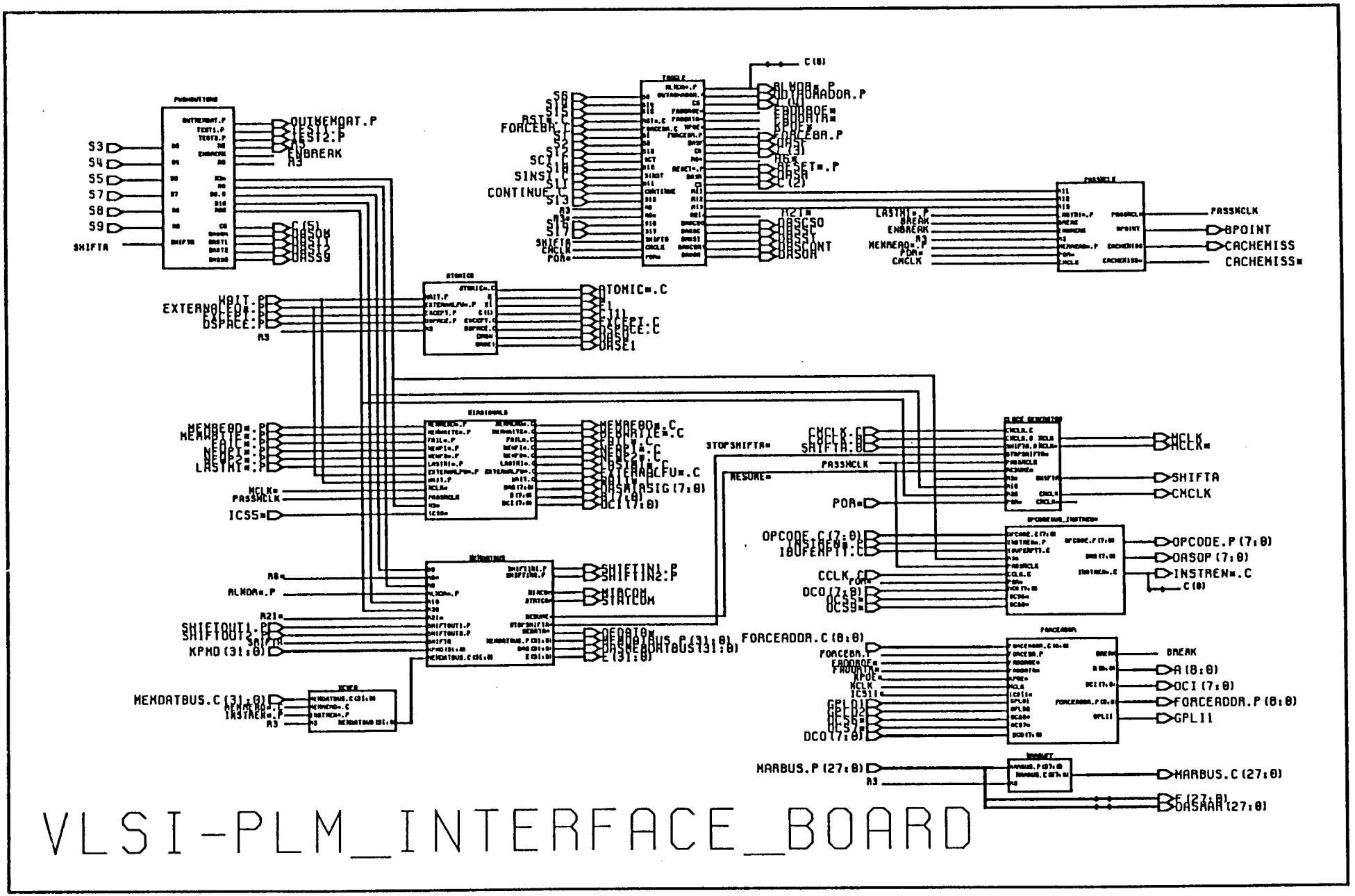

Figure 17 Interface Logic Schematic

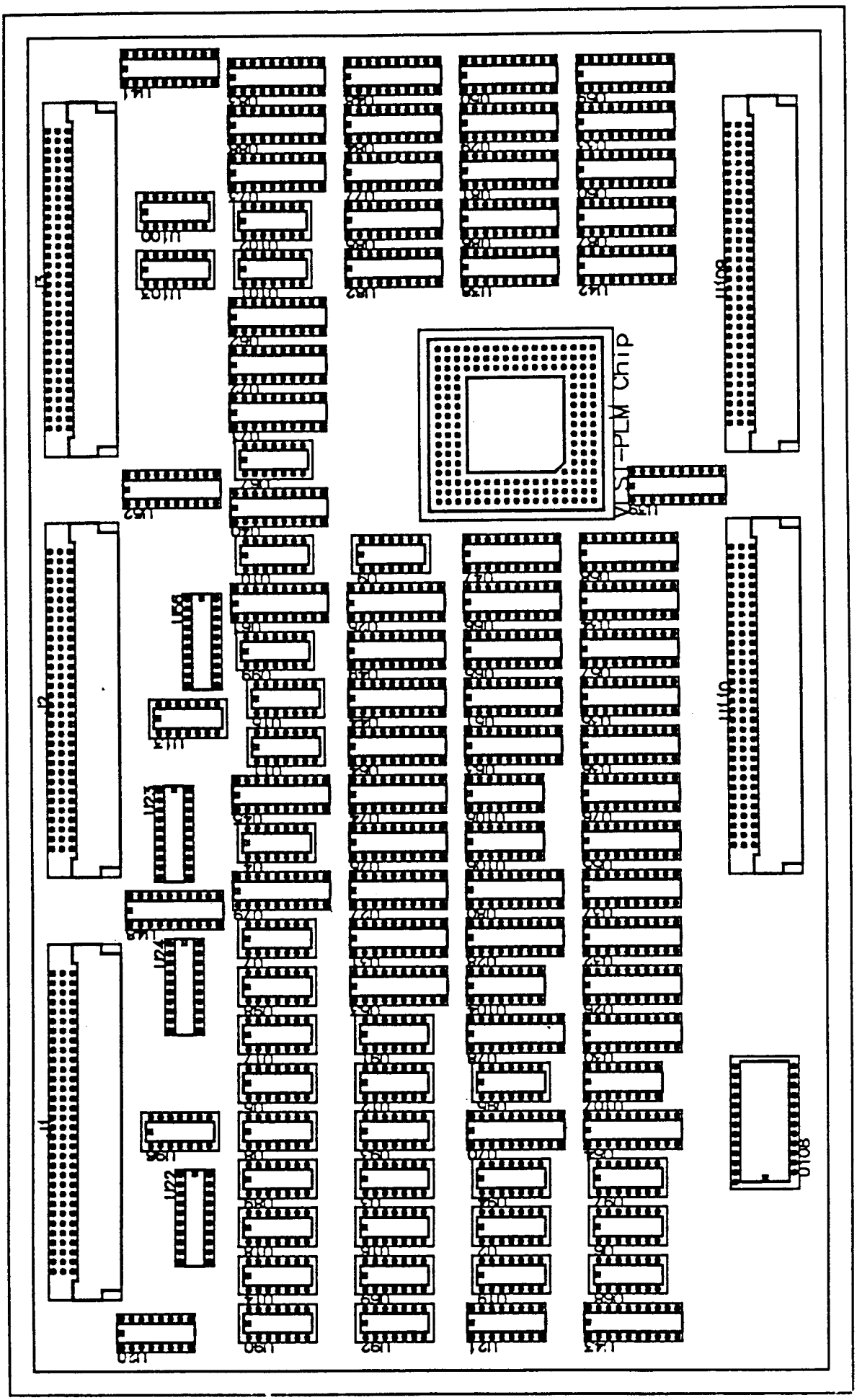

Figure 18 VLSI-PLM PC Board Layout

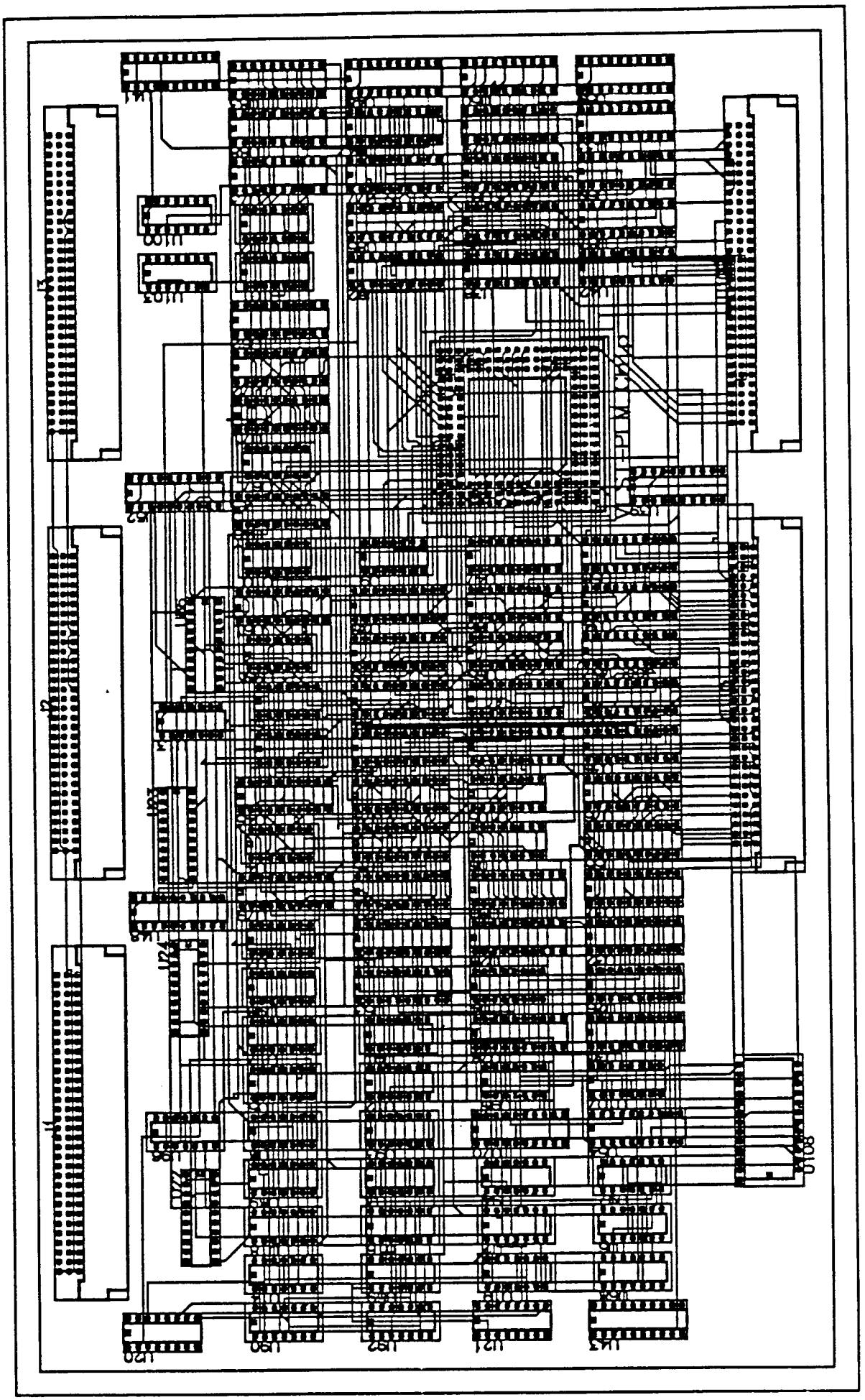

Figure 19 Routed VLSI-PLM PC Board

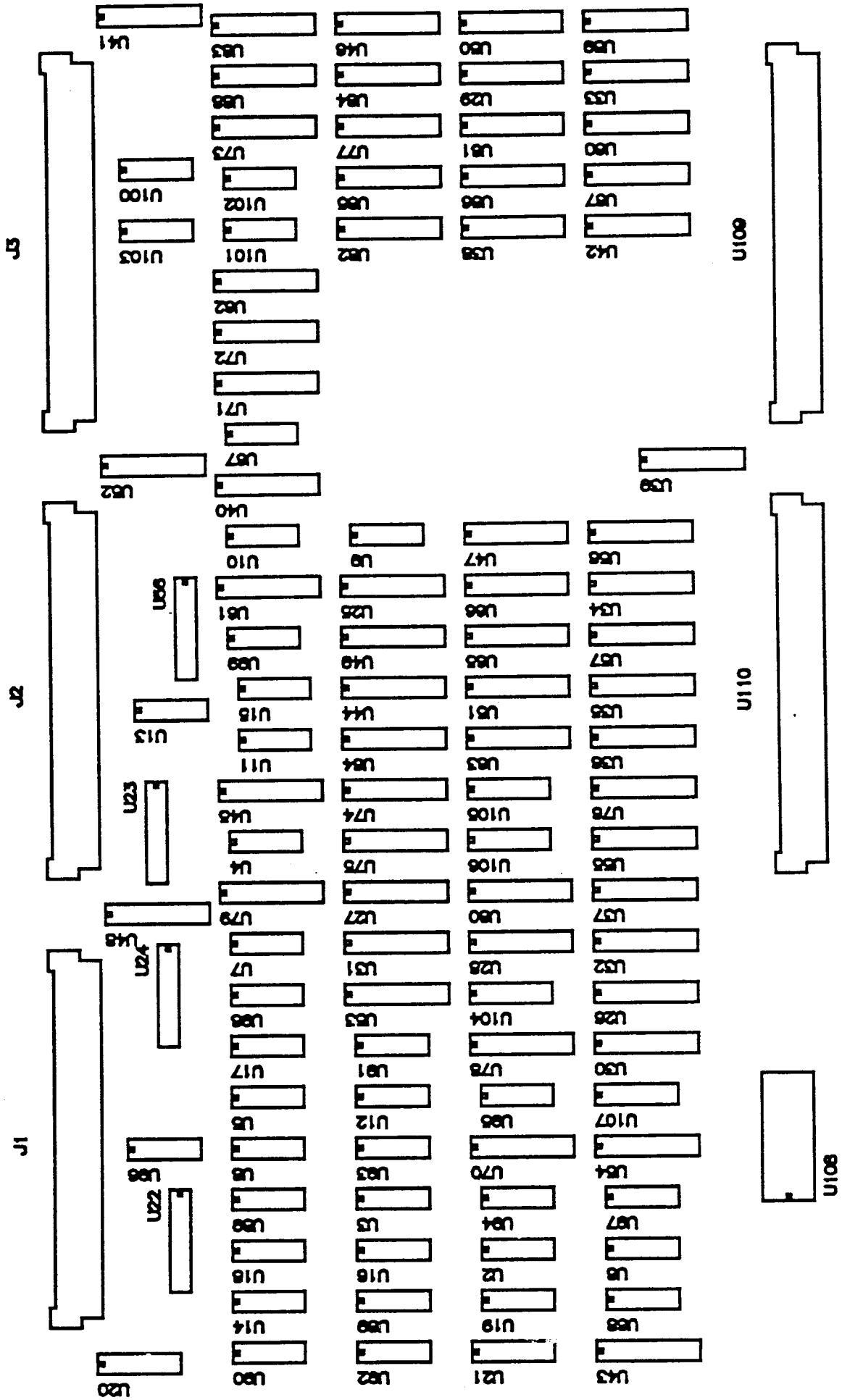

Figure VLSI-C Board Artwork 1

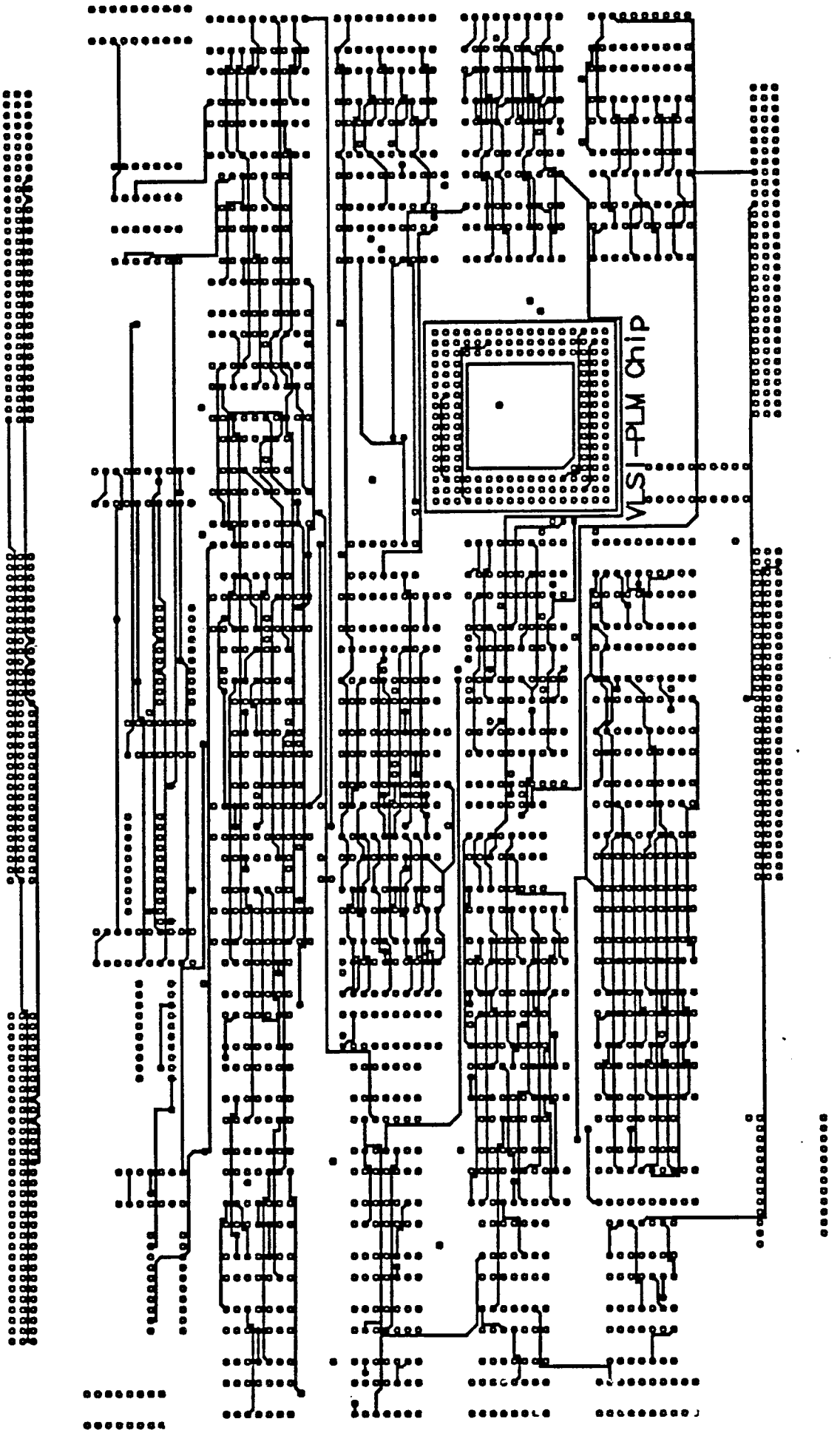

Figure 21 VLSI-PLM PC Board Artwork 2

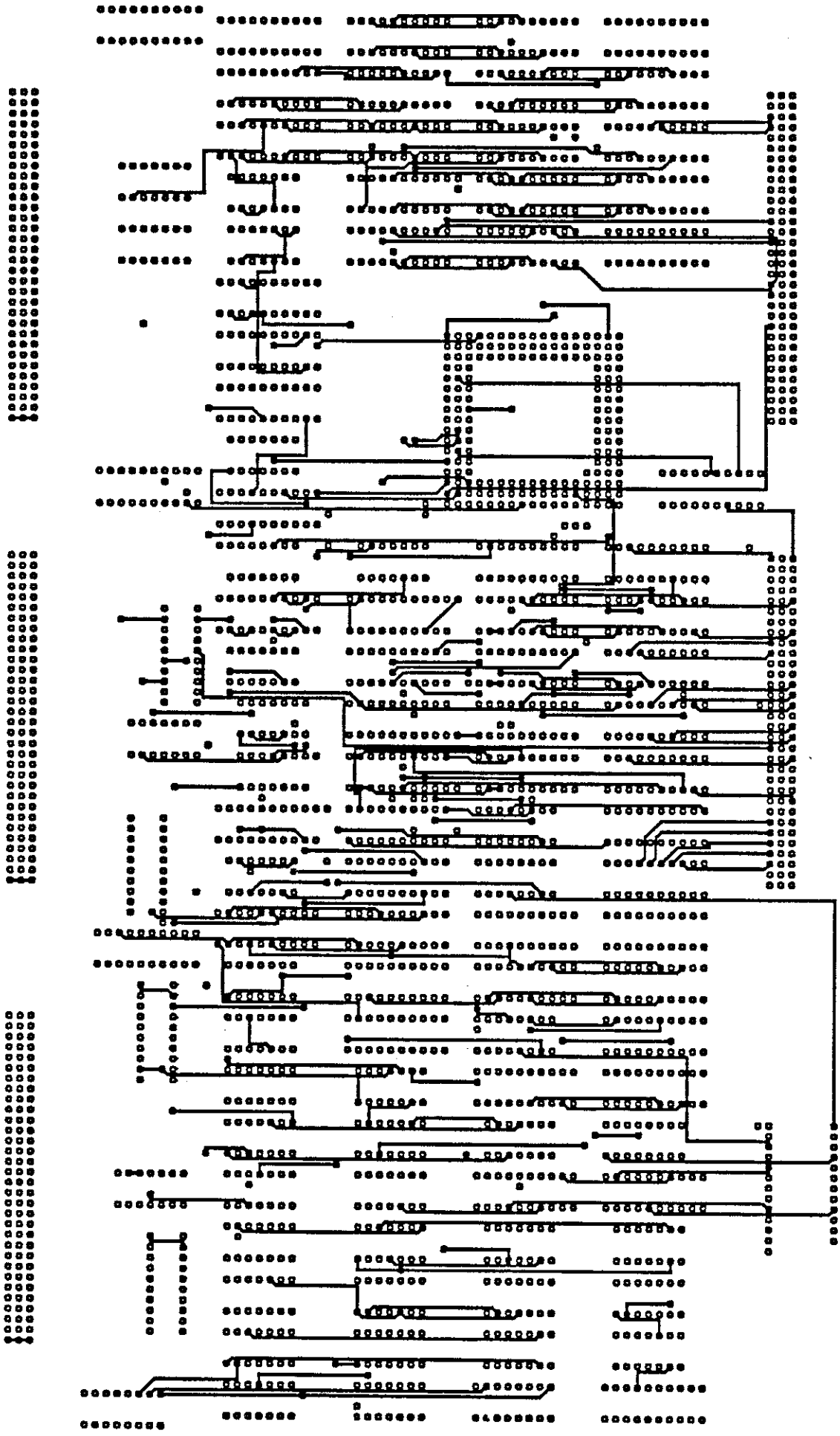

Figure 22 VLSI-PLM PC Board Artwork 3

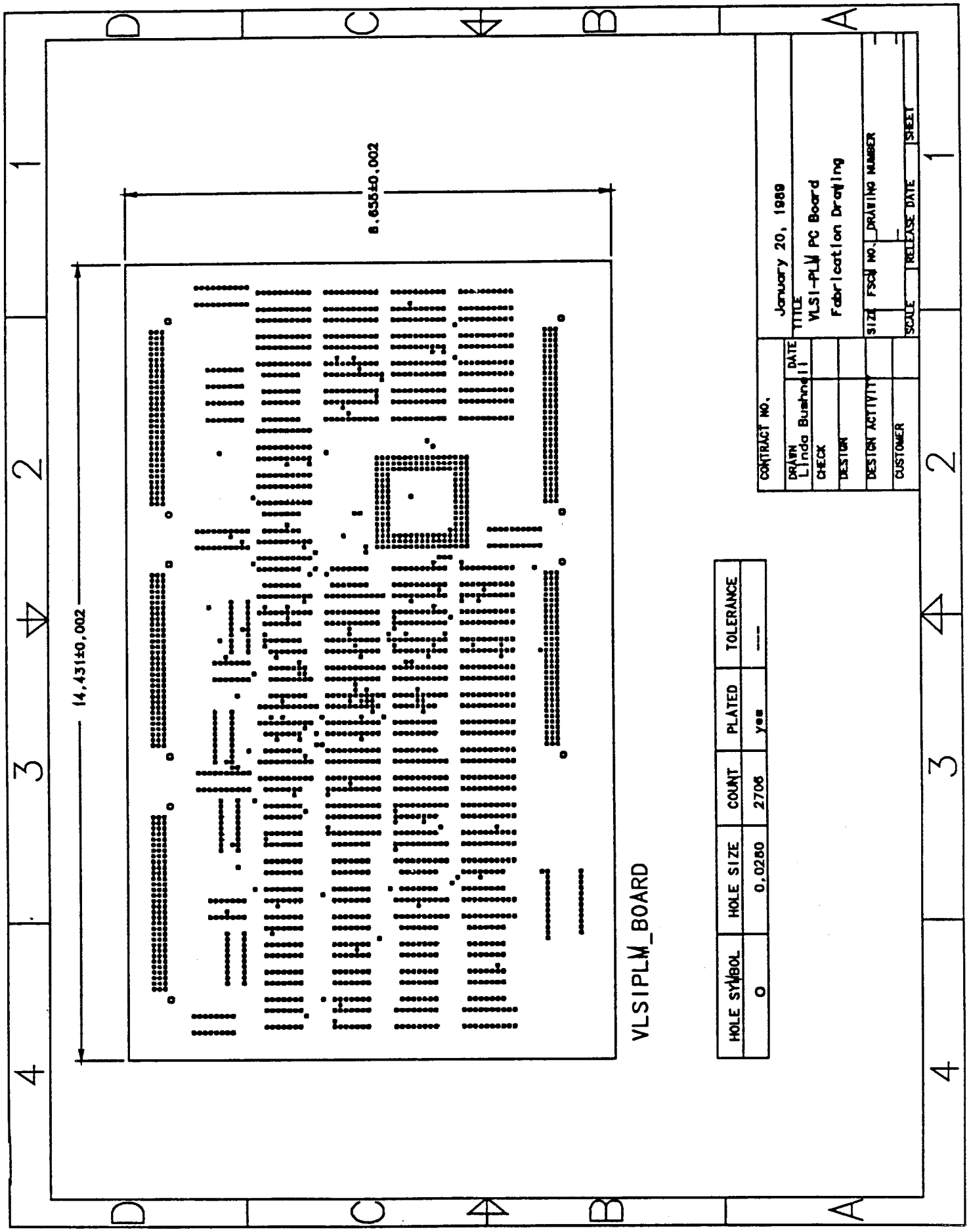

Figure 23 VLSI-PLM PC Board Fabrication Drawing

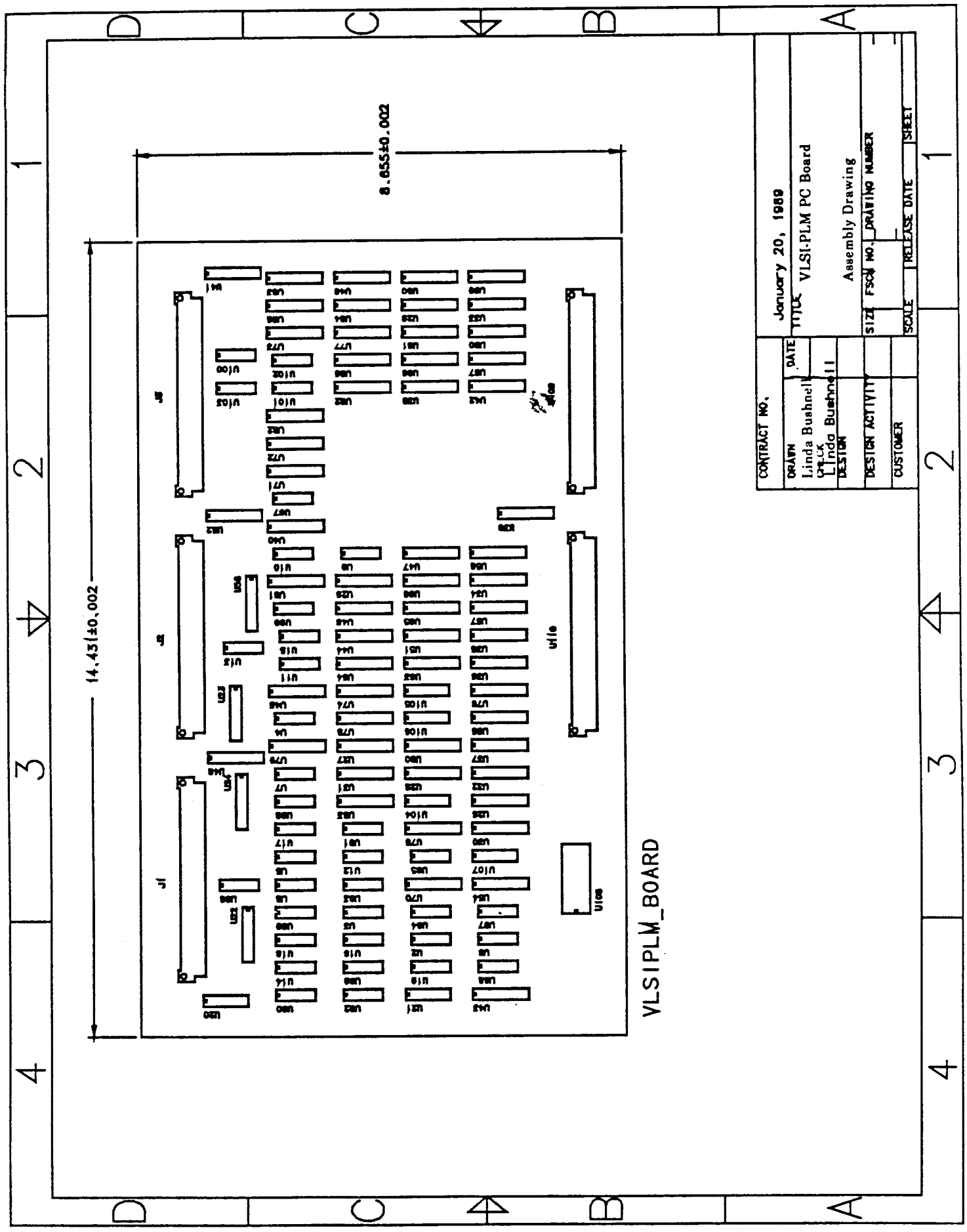

Figure 24 VLSI-PLM PC Board Assembly Drawing